

A Comparison of the Efficiencies of Code Inspections in Software Development and Maintenance

Liguo Yu and Robert P. Batzinger
Computer Science and Informatics
Indiana University South Bend
1700 Mishawaka Ave. P.O. Box 7111
South Bend, IN 46634, USA
{ligyu, rbatzing}@iusb.edu

Srini Ramaswamy
Computer Science Department
University of Arkansas at Little Rock
2801 S. University Avenue
Little Rock, AR 72204, USA
sramaswamy@ualr.edu

Abstract

Inspection is one of the most common sorts of review practices in software projects. However, there are some controversial reports about the efficiencies of software inspections. In this paper, we perform an empirical study to analyze the efficiencies of code inspections in both software development and software maintenance. The study is performed on 650 NASA SEL inspection records. Similar results are found for both the inspections of the original code in software development and the modified code in software maintenance: (1) the efficiency of an inspection meeting is not linearly dependent on the number of inspectors; (2) preparation time and inspection time play critical roles in determining the efficiency of an inspection meeting.

1. Introduction

Inspection in software engineering, refers to peer review of a work product to look for defects using a well defined process [1]. The goal of the inspection is for all the inspectors to reach consensus on a work product and approve it for use. Commonly inspected work products include software requirement specification, design specification, source code, and test plan. In an inspection meeting, a team of several members is gathered. First, each inspector prepares for the meeting by reading the work product and noting the defects. Then, in the meeting, each noted defect is discussed and the conclusion needs to be accepted by all members before it can be formally recorded. Therefore, several factors are related to the efficiencies of an inspection meeting, such as the number of

inspectors, the preparation time, the inspection time, and so on.

A code inspection (code review) is a special kind of inspection in which the team examines the source code and determines any defects in it. A code inspection could be applied in both software development and software maintenance. In software development, a code inspection can detect parts of code that do not properly implement the requirement, that do not function per the design specification, or that are correct but could be improved. In software maintenance, a code inspection can be used to assure the modifications to the source code meet the change requirement and do not introduce regression faults to the system.

Several researchers have studied the efficiencies of code inspection [2] [3] [4] [5]. However, most published results are controversial, with two questions still largely unclear: (1) does the number of inspectors affect the efficiency of an inspection; (2) does the preparation time affect the efficiency of an inspection? For example, in [6], the study shows that preparation time affects the efficiency of an inspection, while in [7], the study shows no such relationship.

To our knowledge, most of the research reported for code inspections is performed on software development to examine the original code. On the other hand, code inspection is also widely used in software maintenance to examine the modified code. This paper makes a contribution to better understanding of the efficiencies of an inspection by using empirical methods to compare the factors that can affect the efficiencies of code inspection in both software development and maintenance.

The remainder of this paper is organized as follows. Section 2 describes the data used in this study. Section

3 presents our empirical study. Our conclusions are in Section 4.

2. Data description

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC). It is created to investigate the effectiveness of software engineering technologies and processes. Data used in this study was collected from about 200 software projects in Goddard Space Flight Center (GSFC) and Flight Dynamic Division (FDD). Data in SEL are submitted on forms by managers, developers, maintainers and testers using either on-line templates or paper forms. Once the forms have been submitted, the SEL data librarian uses software tools to extract the data and load them into the SEL database. Because of the research purpose, the data is well organized, characterized, and stored.

The NASA/SEL dataset used in this study are provided by Data & Analysis Center for Software [8]. It includes 650 records of code inspections. Each record includes complete data about the inspection meeting. The data we studied are divided into two groups, the development group and the maintenance group, which contains the inspection records in software development and in software maintenance respectively. Table 1 shows the number of inspection records in the two groups.

Table 1. The inspection records

Group	Development	Maintenance
Inspection target	Original code	Modified code
Number of records	422	228

For each inspection record, we collect four measures, as described in Table 2. In software development, SLOC is the number of lines of original code inspected; in software maintenance, SLOC is the number of lines of modified code inspected.

It should be noted that there is one difference between our study and most other studies. In most other studies, the efficiency of an inspection is represented with the number of defects found, while in our study it is represented with SLOC—the number of lines of code inspected.

3. The empirical study

Table 3 shows the average preparation_time, inspection_time, n_inspectors, and SLOC in two groups of data.

Table 2. Collected measures

Measure	Description
Preparation_time	The time to prepare for an inspection meeting measured in hours.
Inspection_time	The length of an inspection meeting measured in hours.
N_inspectors	The number of inspectors in an inspection meeting.
SLOC	Number of lines of source code inspected in an inspection meeting.

Table 3. The average collected measures

Inspection Target	Original Code	Modified code
Preparation_time	1.65	1.05
Inspection_time	0.61	0.36
N_inspectors	2.60	1.89
SLOC	357.84	426.46

As mentioned before, in this study, we use SLOC to represent the efficiency of an inspection meeting. Intuitively, we would expect to find the number of lines of code inspected increases with increases in preparation_time, inspection_time, and n_inspectors. In more detail, we tested the following six null hypotheses:

- *H01: There is no linear relationship between the SLOC and the preparation_time in original code inspection.*
- *H02: There is no linear relationship between the SLOC and the inspection_time in original code inspection.*
- *H03: There is no linear relationship between the SLOC and the n_inspectors in original code inspection.*
- *H04: There is no linear relationship between the SLOC and the preparation_time in modified code inspection.*
- *H05: There is no linear relationship between the SLOC and the inspection_time in modified code inspection.*

- *H06: There is no linear relationship between the SLOC and the n_inspectors in modified code inspection.*

In these tests, SLOC is the dependent variable Y, preparation_time, inspection_time, and n_inspectors are identified as independent variable X. To test these hypotheses, we would need to calculate the correlation, which summarizes the strength of the relationship between the two variables X and Y. Several different correlation coefficients have been put forward, including Pearson's correlation coefficient and Spearman's rank correlation coefficient [9]. For Pearson's correlation coefficient to be valid, both variables X and Y need to be normally distributed. However, it is unlikely that the data we gathered for either X or Y is normally distributed. Therefore, we use Spearman's rank correlation coefficient. If the rank correlation coefficient proves to be statistically significant at, say, the 0.05 level, we will reject the null hypothesis.

Table 4 and Table 5 show the correlations between SLOC and other measures and the corresponding p-values in original code inspection and modified code inspection respectively. Because the p-values for H01, H02, H04, and H05 are all significant at 0.01 level, we reject these four null hypotheses and conclude:

- There is positive linear relationship between the SLOC and the preparation_time in original code inspection.
- There is positive linear relationship between the SLOC and the inspection_time in original code inspection..
- There is positive linear relationship between the SLOC and the preparation_time in modified code inspection.
- There is positive linear relationship between the SLOC and the inspection_time in modified code inspection.

Table 4. The correlations between SLOC and other measures in original code inspection

Measure	Preparation_time	Inspection_time	N_inspectors
Correlation coefficient	0.448	0.255	0.082
P-value	<0.01	<0.01	0.094

Table 5. The correlations between SLOC and other measures in modified code inspection

Measure	Preparation_time	Inspection_time	N_inspectors
Correlation coefficient	0.458	0.359	0.097
P-value	<0.01	<0.01	0.143

However, the p-values for H03 and H06 are greater than 0.05, we can not reject the corresponding null hypotheses. This implies that, in both software development and software maintenance, the number of lines of code inspected in one meeting is not significantly linearly dependent on the number of inspectors. In other words, according to our study, adding more inspectors to an inspection meeting does not necessary increase the number of lines of code inspected, because more inspectors means more discussions are needed to get consensus on an issue.

To study the efficiencies of an inspection more deeply, we define two measures: the efficiency of an inspector and the efficiency of a meeting. The *efficiency of an inspector* (Ei) is the number of lines of code inspected per person per hour. The *efficiency of a meeting* (Em) is the number of lines of code inspected

per hour per team. They are calculated using the following two formulas respectively.

$$E_i = \frac{SLOC}{(preparation_time + inspection_time) * n_inspectors}$$

$$E_m = \frac{SLOC}{preparation_time + inspection_time}$$

First, we study the efficiency of an inspector, Ei. Figure 1 shows the average efficiency of an inspector in meetings that have different number of inspectors in review of original code. Figure 2 shows the average efficiency of an inspector in meetings that have different number of inspectors in review of modified code.

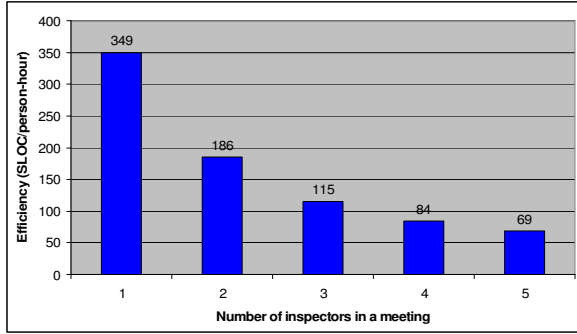


Figure 1. The average efficiency of an inspector (Ei) in review of original code

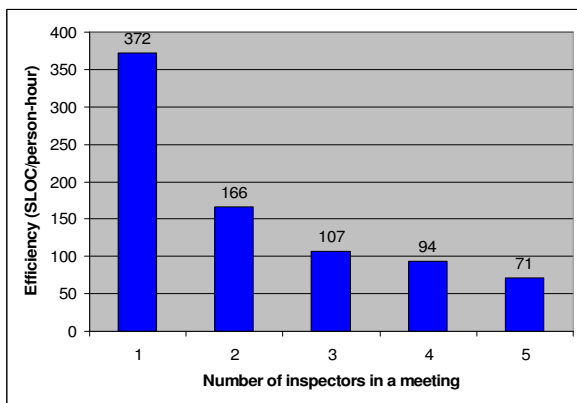


Figure 2. The average efficiency of an inspector (Ei) in review of modified code

Figure 1 and Figure 2 show that the efficiency of an inspector is different for meetings with different number of inspectors: the efficiency of an inspector decreases with the adding of more inspectors. This is not difficult to understand: if two meetings examined the same number of lines of code using the same amount of time (both preparation time and inspection time), the one that has smaller number of inspectors has the higher efficiency E_i . i.e: smaller inspection teams are generally more productive.

Based on Figure 1 and Figure 2, it seems that for inspection meetings with different number of inspectors, the efficiencies of an inspector have similar distribution in development and maintenance. To study their similarities statistically, we test the following hypothesis:

H07: For inspection meetings with different number of inspectors, there is no significant difference between the distribution of the efficiency of an inspector in development and maintenance.

The obvious way to test this hypothesis is to apply the chi-square test. We construct a 2×5 contingency table based on the data shown in Figures 1 and 2. The degree of freedom (DF) for this test is 4 and the chi-square value is 7.344. The corresponding p -value is 0.11. Therefore, we cannot reject the null hypothesis, concluding that for inspection meetings with different number of inspectors, there is no significant difference between the distribution of the efficiency of an inspector in review of original code and in review of modified code.

Next, we study the efficiency of an inspection meeting, E_m . Figure 3 shows the boxplot of the efficiency of an inspection meeting in review of original code with different number of inspectors. (The bold line within the box indicates the median. The box spans the central 50 percent of the data. The lines attached to the box denote the standard range. The circles indicate the data points that are out of the standard range.)

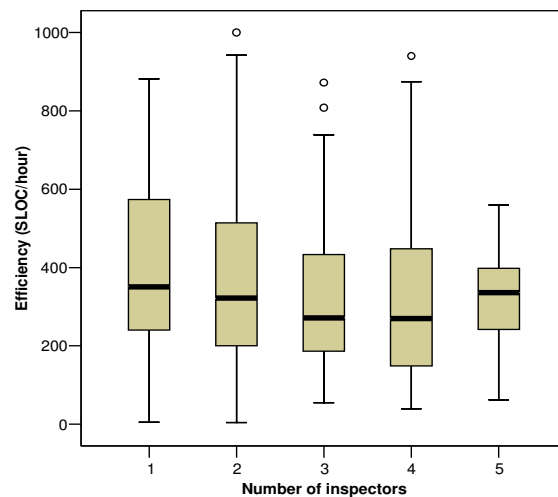


Figure 3. The efficiency of the inspection meeting (Em) in review of original code

Figure 4 shows the boxplot of the efficiency of an inspection meeting in review of modified code with different number of inspectors.

To study whether the number of inspectors can affect the efficiency of an inspection meeting, we tested the following hypotheses:

H08: There is no significant difference between the means of the efficiencies of inspection meetings with different number of inspectors in review of original code.

H09: There is no significant difference between the means of the efficiencies of inspection meetings with different number of inspectors in review of modified code.

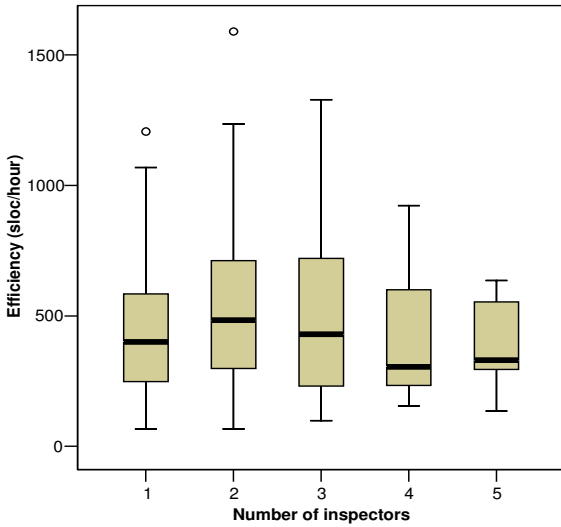


Figure 4. The efficiency of the inspection meeting (E_m) of modified code

To test these hypotheses, we performed two ANOVA tests. The results are shown in Table 6.

In both of the two tests, p -values are larger than 0.05. Therefore, we can not reject these hypotheses. We conclude that there is no significant difference between the means of the efficiencies of inspection meetings with different number of inspectors. This applies to both original code inspection and modified code inspection.

Table 6. The ANOVA test results

Hypothesis	Target code	DF	F value	P-value
$H08$	Original	4	1.872	0.114
$H09$	Modified	4	1.544	0.190

Combing the results shown in Figure 1 through Figure 4, we found that the efficiency of an inspector depends on the number of inspectors; but the efficiency of the inspection meeting does not dependent on the number of inspectors.

Because the number of lines of code inspected in an inspection meeting is not dependent on the number of inspectors, we only consider two factors (preparation_time and inspection_time) to build two models to represent the number of lines of code inspected in a code inspection meeting.

Development model (original code inspection):

$$SLOC = a_1 + a_2 * \text{preparation_time} + a_3 * \text{inspection_time}$$

Maintenance model (modified code inspection):

$$SLOC = a_1 + a_2 * \text{preparation_time} + a_3 * \text{inspection_time}$$

The two models are linear and we use linear regression to estimate their coefficients. Certainly more complicated models could be used, however, to keep the model simple, we prefer using a linear model. Table 7 shows the results of the estimation.

Table 7. Summary of the two linear models

Model	Dependent variable	Independent variable	a_i	P value	R^2	Adjusted R^2
Development	SLOC	constant	174.1	<0.093	0.602	0.595
		preparation_time	675.8	<0.001		
		inspection_time	59.58	<0.001		
Maintenance	SLOC	constant	196.9	0.102	0.785	0.776
		preparation_time	764.4	<0.001		
		inspection_time	30.86	<0.001		

R^2 is the squared multiple correlation coefficient. If the model has a perfect predictability, R^2 equals 1. If a model has no predictive capability, R^2 equals 0. P value is significance. It tells us whether the

corresponding independent variable has statistically significant predictive capability. In both two models, preparation_time and inspection_time have significant predictive capability. This means both

preparation time and inspection time can affect the efficiencies of an inspection meeting.

It worth noting that in both the models, the coefficient of `preparation_time` is greater than the coefficient of `inspection_time`, which means the time spent in preparing for a meeting plays a more critical role in determining the efficiency of an inspection meeting. This indicates that a well prepared meeting is more productive.

4. Conclusions

In this paper, we examined 650 NASA SEL inspection records. Our empirical study clarifies the factors that can affect the efficiency of code inspection. Similar results are found for both the inspection of original code and modified code: (1) the efficiency of an inspection meeting is not linearly dependent on the number of inspectors; (2) both preparation time and inspection time play critical roles in determining the efficiencies of an inspection meeting.

The results presented here not only can help us to understand more clearly about an inspection meeting, but also can provide guidelines for project managers and team leaders to organize inspection meetings and allocate resources.

As with other research, there are threats to the validity of this study. The internal threat is the accuracy of the inspection data. In our study, the data is provided by a third party, we can not assess the accuracy of the data. However, since the data is gathered for the research purpose, we think no reason for questioning its accuracy. The threats to external validity primarily include the subject product and the inspected source code are not representative of other software products. NASA uses his own software process model, which is well known as the waterfall model. To reduce these threats, more studies should be performed on other different inspection records from different organizations that follow different software process in the future.

5. References

- [1] M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development", *IBM Systems Journal*, vol.15, no.3, 1976, pp. 182-211.
- [2] L.G. Votta, "Does Every Inspection Need a Meeting?", *Proceedings of 1st ACM SIGSOFT Symposium on Software Development Engineering*, ACM Press, New York, 1993, pp. 107-114.
- [3] D. Kelly and T. Shepard, "An Experiment to Investigate Interacting versus Nominal Groups in Software Inspection", *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, Toronto, Ontario, Canada, 2003, pp. 122 – 134.
- [4] G. Russell; "Experience with Inspection in Ultralarge-Scale Developments", *IEEE Software*, vol.8, no.1, Jan. 1991, pp. 25-31.
- [5] C. Sauer, D. Jeffery, L. Land, and P. Yetton, "The Effectiveness of Software Development Technical reviews: A Behaviourally Motivated Program of Research", *IEEE Transactions on Software Engineering*, vol. 26, no. 1, January 2000, pp. 1-14.
- [6] F.O. Buck, "Indicators of Quality Inspections." *Technical Report 21.802*, IBM Systems Products Divisions, Kingston, NY, September 1981.
- [7] A. Porter, H. Siy, C. A. Toman, and L. G. Votta, "An experiment to assess the cost-benefits of code inspections in large scale software development." *Proceedings of the 3rd ACM SIGSOFT symposium on Foundations of software engineering*, Washington, D.C., USA, 1995, pp. 92 – 103.
- [8] The Data and Analysis Center for Software. 2006: <http://iac.dtic.mil/dacs/>
- [9] B. Nolan, *Data Analysis, an Introduction*, Polity Press, Cambridge MA, 1994.